AD-P000 221

A SURVEY AND EVALUATION OF FLIR TARGET DETECTION/SEGMENTATION ALGORITHMS

B. J. Schachter

Westinghouse Electric Corporation
Systems Development Divisions
Box 746, Mail Stop 440
Baltimore, Maryland 21203

ABSTRACT

A study was conducted at Westinghouse and the University of Maryland to survey and evaluate FLIR target detection/segmentation algorithms. The study was conducted in software on a data base of 50 FLIR images supplied by the three branches of the armed services. The study concluded that three techniques (double-window filters, spoke filters, and border followers) perform fairly well and are implementable in real-time hardware.

INTRODUCTION

This report describes results of a study conducted both at Westinghouse and the University of Maryland under the DARPA Image Understanding Program. One objective of the study is to survey and evaluate FLIR target detection/segmentation algorithms. Candidate algorithms were chosen from those developed at Westinghouse, the University of Maryland and other academic, governmental, and industrial organizations. Algorithms from the initial list were evaluated over a common data base if and only if: they passed certain preliminary tests, performed well in previous studies, or were claimed to perform exceptionally well by their authors.

Each algorithm was tested by its author's organization on the organization's own computer facility. The study had one rule: "No algorithmic parameters could be changed by human intervention during the evidential run through the data base.*" Refinements could be made to the software before this final pass through the data base.

DATA BASE COMPILATION

A data base of 50, 128 × 128 pixel FLIR images was compiled. Several images from the data base are shown in Figure 1. The sources of the images were four larger data bases prepared by the three branches of the armed services. Twenty of the 50 images were constructed from 64 × 64 pixel target images which were repeated four times by reflecting them about horizontal and vertical axes (e.g., see Figure 1b). These were used to test the algorithms' sensitivity to the orientation of targets.

The data base was compiled with a goal of diversity. Often results are reported in the literature in which an algorithm is tested on a very small collection of similar images. This allows an algorithm to be finely tuned to the ensemble statistics. In compiling a diverse data base, it was hoped to simulate realistic conditions in which little a priori information is available.

EVALUATION APPROACH

The data base was carefully hand segmented to obtain the centroid (C_I, C_J) and vertical (R_I) and horizontal (R_J) radii of each of the targets (Figure 2). This hand segmentation was a cooperative effort of several persons who had knowledge of the data base contents. All algorithms were required to use this same format for output. That is, for each detected target, in each image, the output is an estimate: ($\hat{C}_I$, $\hat{C}_J$, $\hat{R}_I$, $\hat{R}_J$).

A target is said to correctly *detected* iff:

$$\{|C_I - \hat{C}_I| \leq \tfrac{1}{2}R_I + \tfrac{1}{2} \text{ and } |C_J - \hat{C}_J| \leq \tfrac{1}{2}R_I + \tfrac{1}{2}\}, \quad (1)$$

where all measures are in image picture elements (pixels). A *false alarm* is any detection outside this prescribed region. Extra detections inside the prescribed region are each scored as 1/3 false alarm.

With reference to Figure 3, let B denote the rectilinearly oriented box hat enclosing a target as determined by hand segmentation. Let $\hat{B}$ denote its estimate as output by a computer program. Segmentation accuracy is estimated by:

$$A = \frac{|B \cap \hat{B}|}{|B \cup \hat{B}|} \quad (2)$$

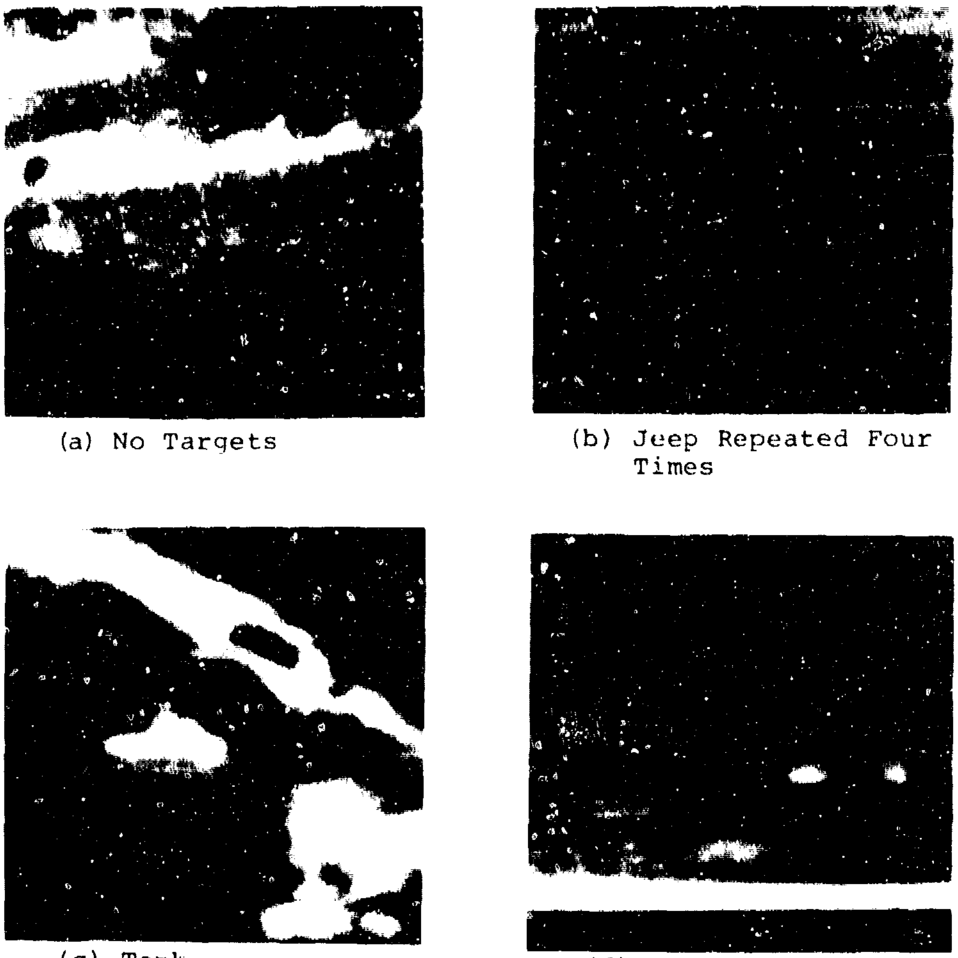
or the common area divided by the total area of the figures, averaged over detected targets. Note: $0 \leq A \leq 1$.

TESTS PERFORMED BY COMPUTER SIMULATION

One goal of this project is to determine the suitability of various hardware implementations of the candidate algorithms. This entails an evaluation of both performance and implementability. However, the tests discussed here took place only in software and did not take advantage of the additional information which would be available to a real-time target acquisition system, such as range to center of field of view. This point will be expanded upon below.

A target acquisition system typically consists of a number of pipelined stages. The first stage accepts images from the FLIR and does some preprocessing. This Preprocessor usually takes the form of one or more local filters which reduce noise, extract local features, and/or increase the contrast between targets and background.

* Note: Image polarity was allowed as an input.



(a) No Targets

(b) Jeep Repeated Four Times

(c) Tank

(d) Tank, APC

Figure 1. Four Images From Compiled FLIR Data Base

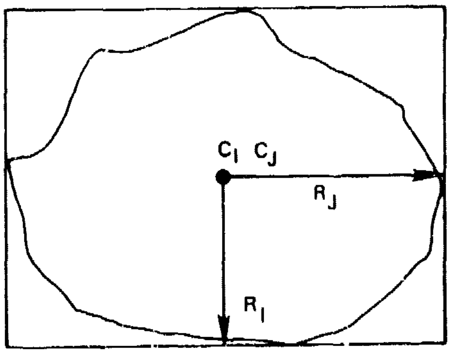


Figure 2. Illustration of Target Location and Extent Parameters

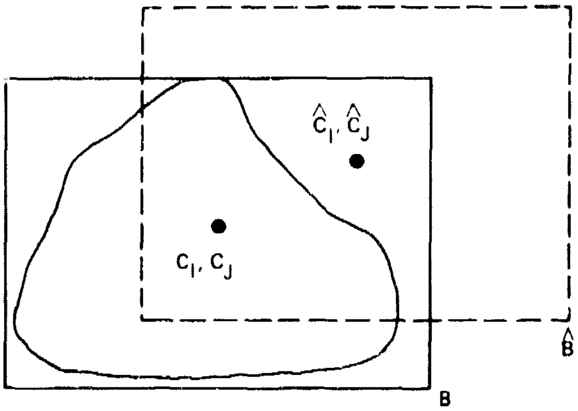


Figure 3. Illustration of Relation between Boxes Enclosing Actual and Detected Targets

The second stage is the Detector, which locates blobs it suspects of being targets. The centroids of these candidate targets are then passed to a Segmentor and/or Feature Extractor. This third stage either determines the borders of the detected blobs or extracts features to describe them. This information is passed to the fourth stage, called the Classifier. This final stage classifies potential targets, prioritizes them, determines confidence probabilities, and outputs this data along with target locations.

A real-time system mounted on a moving vehicle should take advantage of the following types of additional information which are usually available.

(1) *Range* - The range to the center of the field of view will usually be known. This will permit calculation of expected target size in pixels at that location. Size in pixels at other locations in the field of view can be estimated from assumptions or knowledge of terrain geometry.

(2) *History* - Past values of any computed parameters can be stored. (For example, the location of a detected target on past frames can be used to predict current location.)

(2) *Control Loops* - Feedback loops can be set up between any stages. (For example, the output of the Classifier can be used to control thresholds in the Detector.)

If a real-time system fits the model described, then separate circuit boards might be used for each stage. Boards implementing different algorithms could be interchanged, and the algorithms tested on large quantities of data. If board pin-outs and protocols are standardized, then even boards developed by different companies could be interchanged and tested in a competitive manner. No such standardization now exists. Furthermore, in current real systems, the separation in functions between the various stages is often not clearly defined. An algorithm implemented on a board may combine parts of several stages. Even the basic hardware organization may be different from that discussed above, with some stages operating in parallel or in a different order than indicated. There is always a tradeoff in performance between the various stages, since it is only the output of the final stage which really matters. For example, it is acceptable for a detector to operate at a high false alarm rate if a classifier will later cull out counterfeit targets. This is an important qualification for a study such as this, where individual algorithms are being tested, not entire systems. We will deal in part with this concern by plotting detection rate vs. false alarm rate, though our results are not as comprehensive as may be desired.

CLUSTERING IN MEASURE SPACE

The standard method of segmenting an image is by gray level thresholding. Here the classes correspond to gray level ranges, e.g., "light = hot" and "dark = cool". Since these ranges are not known in advance, they must be determined by examining the gray level histogram and looking for peaks (one dimensional clusters), and choosing thresholds (one dimensional decision surfaces) that separate the peaks.

A number of investigators have suggested that multidimensional feature space should also be useful for segmenting complex gray scale images. A variety of features may be defined over a neighborhood set about a pixel, e.g., mean, median, variance, commonality, total variation. This approach could be employed when a single feature, such as gray level, is not adequate for segmentation because the given image contains a number of textured regions whose gray level ranges overlap.

Initial work at Westinghouse has indicated that thresholding by cluster detection in histograms is not adequate for separating targets from background. Gray level target and background clusters are often not separable, i.e., their probability densities overlap. Likewise, the response of local operators tends to be rather variable, not yielding well defined clusters. The basic weakness of segmentation schemes which use only local feature values is that they attempt to classify image parts without regard to their relative positions in the image. It should not surprise us that any approach which does not take spatial contiguity fully into account fails much of the time.

ALGORITHMS SELECTED

The segmentation algorithms investigated in our study are those that make use not only of similarity but also proximity. The candidate algorithms have been classified into six groups:

(1) Double Window Filters
(2) Spoke Filters
(3) Border Followers
(4) Relaxation Algorithms
(5) Pyramid Approaches
(6) Mode Seekers

Each of these approaches will be described below. At least one method of each type will be tested on the assembled data base.

DOUBLE WINDOW FILTERS

A double window filter slides two non-overlapping windows over an image. The windows are both commonly rectangles, with one surrounding the other (Figure 4). The intensities within the inner window are viewed as samples of a random variable X and those of the outer window as samples of a random variable Y. The objective is to determine if the inner window surrounds a target, while the outer window contains background clutter (Figure 5). Since little a priori information is available about target and background statistics, some assumptions are usually made before the problem is formulated and solved. One way to pose the problem is in the language of statistical hypothesis. Doing so usually involves choosing distributions to model the behaviors of X and Y. Results, of course, are valid only if the chosen distributions correctly describe the experimental situation, namely that they provide the correct statistical model. Although *simple* statistical hypotheses concern only the parameters of assumed or known distributions, it is also possible to define *composite* hypotheses which also concern the fundamental forms of the distributions themselves. In either case, to construct a criterion for testing a given statistical hypothesis requires the formulation of an alternate hypothesis. Symbolically, we will let H_0 stand for the null hypothesis and H_A stand for the alternate hypothesis.

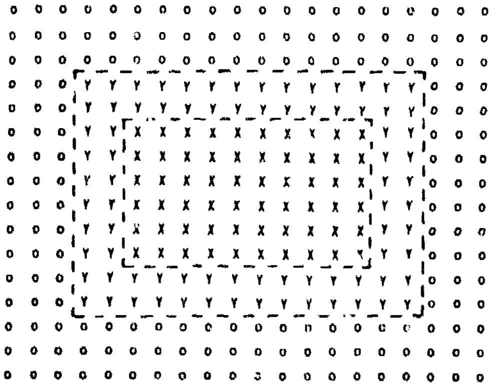


Figure 4. Window Geometry

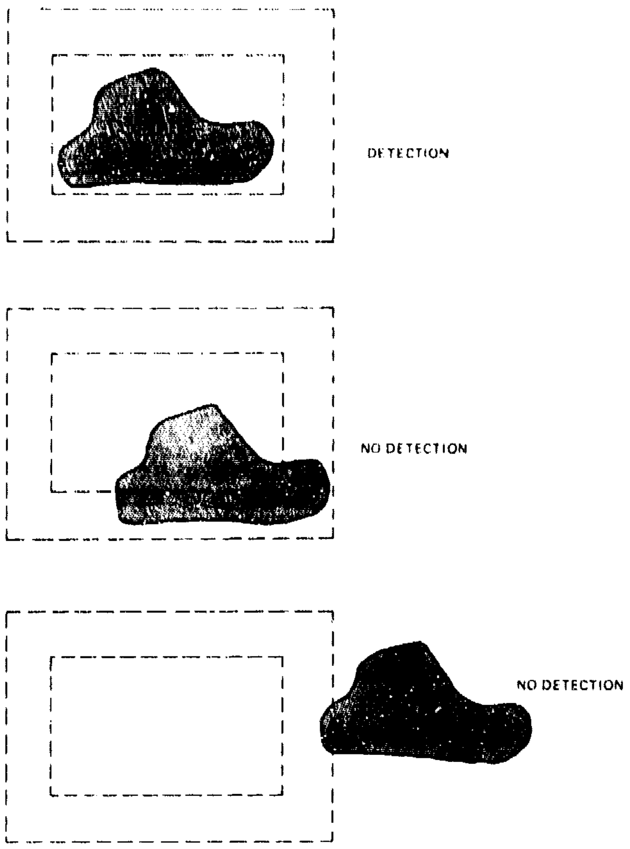


Figure 5. Three cases of target/filter relationship: (a) target completely within inner window, (b) target partially within inner window, (c) target outside inner window.

A typical simple hypothesis is as follows:

$X_1, \ldots, X_m$ is an independent random sample of a normal random variable X with mean μ_X and unknown variance σ^2_X

$Y_1, \ldots, Y_n$ is an independent random sample of a normal random variable Y with mean μ_Y and unknown variance σ^2_Y

$$H_o: \mu_X = \mu_Y$$
$$H_A: \mu_X \neq \mu_Y$$

This test can be performed in terms of a quantity T which has a t-distribution with m + n - 2 degrees of freedom [18]:

where

$$T = \frac{k(\overline{X} - \overline{Y})}{\sqrt{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2}}$$

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i, \quad \overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i. \qquad (3)$$

H_o is rejected at a significance level α if and only if $t \geq c$, where t is the experimental value of T. If, for instance, $n = 10$, $m = 6$, and $\alpha = 0.05$, then $c = 2.145$.

If targets are always known to be hotter (i.e., of higher intensity level) than the background, the test can be reformulated as:

$$H_o: \mu_x = \mu_Y$$
$$H_A: \mu_x < \mu_Y \qquad (4)$$

This approach and more complex formulations are conventionally used by Westinghouse in detection of targets in radar. A number of companies are using it for target detection in FLIR imagery.

Texas Instruments [TI] and Ford Aerospace [FA] have developed filters of this type. TI [2] uses a metric of the form:

$$C = \frac{(\overline{X} - \overline{Y})^2 + \hat{\sigma}_X^2}{\hat{\sigma}_Y} \qquad (5)$$

FA [1] divides the outer window into N subregions. They use a metric of the form:

$$C = \sum_{k=1}^{N} (I_X - I_Y(k)), \qquad (6)$$

where I_x is the mean of inner window pixels exceeding a specified threshold and $I_Y(k)$ is the mean of outer window pixels in subregion k exceeding the threshold. Both TI and FA use range for the control of window size. Neither referenced paper provides a statistical model for the experimental design.

Woolfson of Westingthouse sees no justification in assuming normality or any other statistical distribution. He suggests an approach based on estimates of the probability density functions of random variables X and Y (as obtained from their sample values). A computer program was written to test this concept. The upper three bits of sample intensity levels were used to estimate probability density functions f_X and f_Y. Since three bits were used, these estimates in effect were quantized to eight entries each; i.e., $\hat{f}_X = (\hat{f}_X(1), \ldots, \hat{f}_X(8))$, $\hat{f}_Y = (\hat{f}_Y(1), \ldots, \hat{f}_Y(8))$. A target is detected if and only if the maximum intensity level in the inner window exceeds that of the outer window and

$$\sum_{i=1}^{8} |\hat{f}_X(i) - \hat{f}_Y(i)|$$

exceeds a given threshold. Since range was not available for these experiments, five filters were used in order of decreasing size, with a detection occurring upon first exceeding the threshold. Since several detections often occur over nearby pixels, detections were spatially clustered and replaced by their cluster centroid. The associated window widths and heights were averaged to form an estimate of target extent. Results are shown in Figures 6 and 7. The method works fairly well. It has difficulty with targets near image borders, but this is not a serious problem in an actual target acquisition system where the input imagery is typically 875 x 875 pixels in size and targets cover a relatively small area. Since this method uses raw gray levels, rather than edges, it is bound to produce a fairly stable output from frame to frame.

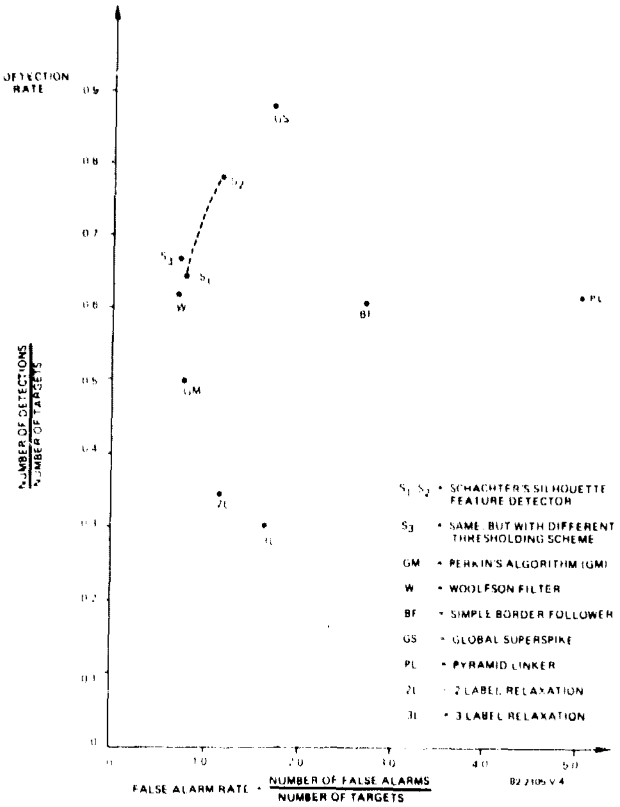


Figure 6. Detection Rate vs False Alarm Rate

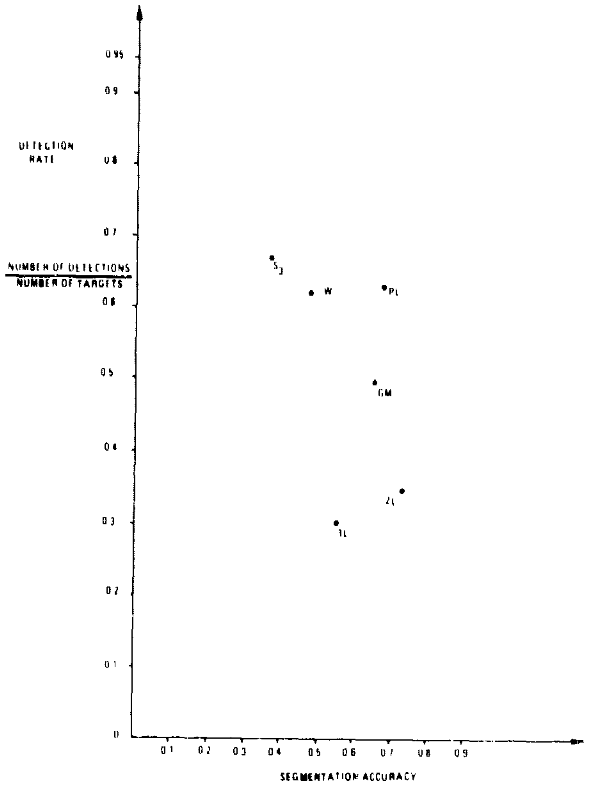


Figure 7. Detection Rate vs Segmentation Accuracy

SPOKE FILTER

Minor and Sklansky's [4] spoke filter is a generalization of the Hough circle detector [5]. The spoke filter uses an 8-spoke digital mask, which can be designed to detect either light or dark blobs, as illustrated in Figure 8. The arrows in the figure define a template. Each detected edge falling within the filter area is examined to determine if its strength exceeds a threshold and matches the direction of the corresponding template element. An 8-bit image size detection buffer is used to store edge matches. Each bit corresponds to one spoke direction. The n-th bit is set at buffer location (x,y) if at least one match is obtained along the n-th spoke when the filter is centered at (x,y). Upon completion of the spoke filtering operation, a 3 x 3 OR filter is convolved with the detection buffer. A detection is then considered obtained at position (x,y) if at least N bits of the (x,y) location of the buffer are set (typically N = 4).

Minor and Sklansky perform segmentation after detection is completed. The segmentor used is a modified version of the University of Maryland's Superslice [3,20,21] algorithm. Superslice views objects as being distinct from their surroundings by the presence of edges at their boundary. Superslice starts by choosing a threshold to segment the image into a set of connected components. It then extracts an edge map from the image. Finally, it measures the percent of each component's border which coincides with the edge map. An extracted blob is one which produces high edge-border coincidence. A number of thresholds are usually tried before good edge-border coincidence is obtained. Different thresholds may be required for extracting different objects in the same scene. Minor and Sklansky's version of Superslice also uses edge direction in the measure of coincidence.

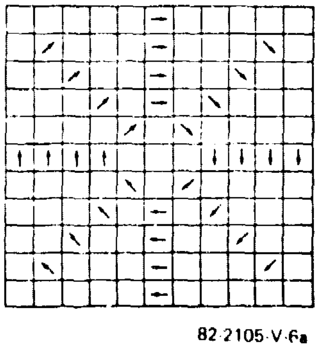


Figure 8a. Spoke Filter for Bright Objects on a Dark Background (From an Unpublished Version of [4])

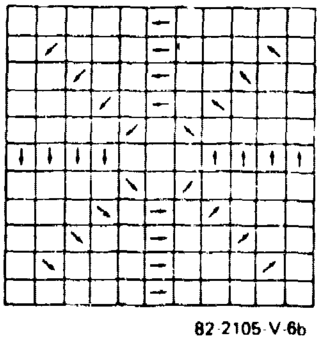


Figure 8b. Spoke Filter for Dark Objects on a Bright Background

A group at Hughes [6] briefly describes a method which tests each pixel in a region of interest for boundedness by edges in eight directions. A pixel is labeled to be interior to some boundary if it is bounded by edges in six of the eight directions. The labeled image is then thresholded to extract a target interior. If $\bar{Y} \le \bar{X}$ then pixels with intensity greater than threshold C are selected, otherwise pixels with intensity less than C are chosen. Finally, a thinning and filling operation is used to remove small holes and smooth the segmented region. The Hughes report supplies no statistical model from which their thresholding equation is derived.

The author developed another algorithm based upon the basic spoke filtering philosophy. The algorithm uses sets of 3x3 pixel templates - with each template designed to match a section of a particular object's silhouette. The operation of the algorithm will be described by example.

First, suppose that an algorithm is to be designed to detect octagons of known polarity. A 4-bit image size buffer (initially zeroed) will be used to record detections. The algorithm will start by scanning across each of the rows of an image, applying a 3x3 vertical edge detector at each point visited. When a left blob edge is detected, its position will be noted (Figure 9a). The scan will continue until a right blob edge is detected (Figure 9b). At this time, a 0001_2 will be OR'd with the contents of the buffer at a location corresponding to the midpoint between detected edges (figure 9c). This process is repeated upon the image columns and in the two diagonal directions using the markers shown in Figure 10a.

At the completion of the scans in the four directions, we would expect to find a 1111_2 at buffer locations corresponding to centers of octagons in the original image. Since this may not always occur, due to the discrete nature of the image geometry, a 3×3 OR filter is convolved with the detection buffer.

Now suppose that we want to detect military vehicles. Instead of applying edge detectors in the four scan directions, feature detectors will be applied which correspond to shapes located around the sought vehicles' silhouettes. Also, the diagonal scan directions should not be exactly 45 degrees to image rows, but rather should be related to the average height to width ratio of targets (Figure 10b).

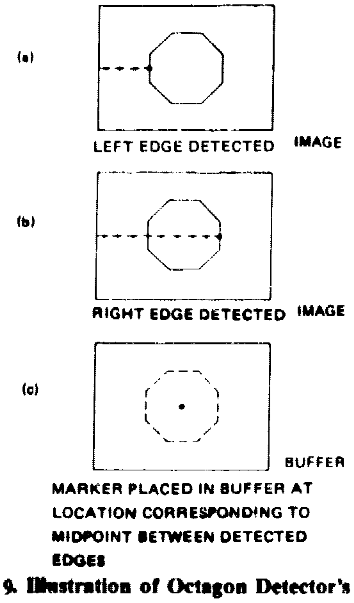

(a)
LEFT EDGE DETECTED
IMAGE
(b)
RIGHT EDGE DETECTED
IMAGE
(c)
BUFFER
MARKER PLACED IN BUFFER AT LOCATION CORRESPONDING TO MIDPOINT BETWEEN DETECTED EDGES

Figure 9. Illustration of Octagon Detector's Scan of an Image Row

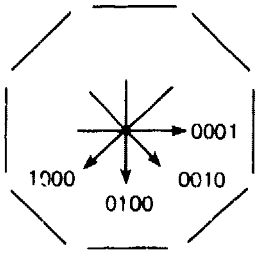


Figure 10a. Templates and Scanning Directions for Octagon Detector

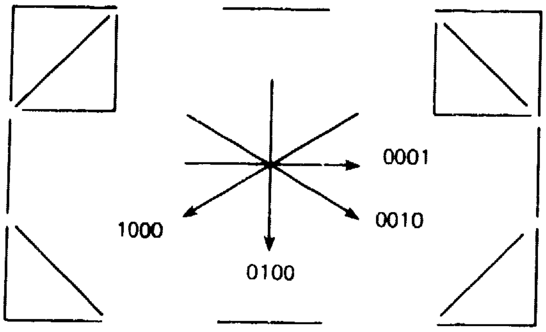


Figure 10b. Templates and Scanning Directions for Military Vehicle Detector (A Succession of 8 Features Represents a Particular Template)

Upon completion of the four scans, a spatial clustering algorithm is applied to detections, with cluster points replaced by their cluster centroid.

The algorithm performs very well as a detector as seen in Figure 6. Detection rate was 67 percent with 0.69 false alarms per target. The algorithm was also evaluated as a segmentor (Figure 7) with the distances between a centroid and the horizontal and vertical feature detections used to estimate target radii. Segmentation accuracy was not very good, with targets usually judged smaller than they actually were.

BORDER FOLLOWING/EDGE BASED SEGMENTATION

A blob outline containing no breaks is a closed curve separating the blob from its background. This curve may pass through the same pixel twice if the blob has a narrow neck near or at this pixel. A number of border following algorithms are described in the image processing literature, e.g., see [7-10]. They start by locating a prominent border point, and continue by visiting adjacent border points (edge elements) in sequence, eventually returning to the starting point. Difficulties arise when an object's boundary is not distinct over its entire length.

Schenker and Cooper [7,9] describe an elaborate algorithm which seeks the most probable blob boundary by conducting an exhaustive search over all boundaries. The "true boundary" is then viewed as the one which maximizes the joint likelihood of the observed data and a hypothesized boundary. Since an exhaustive search proves to be computationally prohibitive, suboptimal search algorithms are also proposed.

Perkins [15], while with General Motors Research Lab, developed an object detection/segmentation algorithm for use in a robot vision system. The algorithm is implemented by a sequence of simple operations. First, an edge map is obtained. Next, uniform intensity regions are extracted by expanding active edge regions, labeling the segmented uniform intensity regions, and then contracting the edge regions. The region with the most points along the image border is assumed to be the background. Finally, foreground region boundary points are visited, the locations of points which are roughly equidistant along a boundary are stored in an array. The performance of Perkins' algorithm as a FLIR target detector is not particularly good (Figure 6). The detection rate over the test data base was only 50 percent. The algorithm as implemented at GM does not use image polarity and, as a consequence, sometimes misses nearby targets, while indicating a detection half-way between them. The performance of the algorithm as a segmentor is excellent (Figure 6).

A simple border follower was programmed and tested at Westinghouse. A detection rate of 61 percent was obtained at a very high false alarm rate (Figure 7).

RELAXATION ALGORITHMS

Rosenfeld et al. [12, 19, 23], Kirby [13], Peleg [14], and others have viewed image segmentation as a graph labeling problem. The proposed approximate, iterative, parallel solutions are called "relaxation methods" which work as follows. First an initial set of labels is assigned to each image node based upon local image properties. The labels at each node are given weights between 0 and 1. This initial set of assignments may be ambiguous or incorrect in spots because of the imperfect nature of the local image measures or noise in the image. An iterative process using information obtained from neighbors is then used to improve the initial decisions. The weights at each node are simultaneously updated at each iteration (typically) based upon the weights at node neighbors during the previous iteration. This has the effect of altering the probabilities initially assigned to noise points to make them more consistent with their surround. The process is stopped when the labeling of all nodes seems to reach a steady state.

To apply 2-label relaxation to segmentation [23], a set of "light" and "dark" probabilities is assigned to image nodes based upon their gray levels. Probabilities at each node are then iteratively adjusted based upon probabilities at neighboring nodes, i.e., light reinforces light and dark reinforces dark. Eventually, the blob pixels should become uniformly light, and the background uniformly dark, so that segmentation is easy. The blob region is extracted by locating connected components. A 3-label scheme uses labels of "object", "background", and "clutter".

The relaxation algorithms tested performed rather poorly as target detectors (Figure 6).

PYRAMID APPROACHES

Let the size of an image be $2^n \times 2^n$ pixels. Reduced resolution versions can be linked with a pyramid data structure. The top (level 0) of the pyramid will be a size 1 × 1 image. The original $2^n \times 2^n$ image will be at the bottom of the pyramid. At level K will be an image of size $2^k \times 2^k$. A number of different schemes have been developed for detecting blobs with pyramids [24].

The pyramid linking scheme of Burt, Hong, and Rosenfeld [28] works as follows. A father-son relationship is defined between nodes of adjacent levels of the pyramid. Each node at level K is the father of a 2 × 2 array of "candidate son" nodes at level K + 1. Likewise, each node at level K is the son of four "candidate father" nodes at level K-1. At each iteration of the pyramid linking algorithm, a node is linked to the "most similar" of its four candidate fathers. A node is then assigned the average gray level of those sons linked to it. The process continues in this manner until a steady state is reached. The links then define trees, and the leaves of "light" trees are target segments. The particular pyramid linking algorithm tested did not perform well as a detector, yielding too many false alarms* (Figure 6).

MODE SEEKERS

There is a class of techniques in which a pixel is iteratively replaced by the average of a selected set of its neighbors. Narayanan and Rosenfeld [29] describe a method that chooses those neighbors for averaging which belong to the same histogram peak as the given pixel. The simplest version of this method chooses those neighbors higher up on the image histogram peak. This tends to move pixel gray levels toward their subpopulation modes. An improved version chooses a neighbor only if there is no significant concavity in the histogram between it and the given pixel. Ideally, upon termination, there will be a two-spiked histogram, with the lighter spike formed from target pixels. Target regions are then obtained by extracting connected components.

A local version of the algorithm performs processing over relatively small image windows in sequence, while the global version (named Global Superspike) uses the histogram for the entire image. The global version was tested since the images in the data base are only 128 × 128 pixels in size. Its detection rate was 88 percent (Figure 6).

EVALUATION

The mode seeker had the highest detection rate of any of the methods tested. Its segmentation accuracy fell in the 65 percent to 75 percent range, as did most of the other methods.

Spoke filters perform very well as detectors. But they have a low segmentation accuracy, and may need to be followed by a separate segmentor such as Superslice.

Double window filters work well as detectors. They can also be used for segmentation if only target extents are required. But preferably, they should be designed to seek out targets of particular sizes if range is available as an input. If targets of several sizes and orientations are sought, then a different filter is required for each. Each filter must vary in size down the image to correct for perspective. This has a multiplicative effect on hardware size and cost.

Simple border followers must be run at a high false alarm rate to yield a reasonable detection rate. They are well suited for segmentation, yielding estimates of target borders. If an entire image is available for processing, the border following can start at the most prominent border point and proceed from there. If an image is to be processed one line at a time (on-the-fly), then the topmost target point must be detected to initialize the follower. This second approach may have difficulty with non-convex targets but is easier to build into hardware.

Border followers and double window filters can be readily implemented in hardware. However, as noted above, proper implementation of the double window filter requires considerable computation power. The spoke filter in its purest form, as proposed by Minor and Sklansky, is not well suited for hardware

* Note: Another pyramid scheme was recently tested at the University of Maryland on this data with much better results.

implementation. It is computationally expensive requiring the repeated access of a large array of edges (or pixels) as the filter sweeps over the image. The algorithm can, however, be reformulated in a number of different ways to meet the requirements of particular computer architectures.

Relaxation algorithms are best implemented in a cellular array architecture having one processor per pixel. No one has ever built a target acquisition system using this architecture; to do so must therefore be viewed as a risky venture. The same holds true for pyramid approaches.

There are two difficulties with implementing the mode seeker. A FLIR image is typically 875 × 875. A mode seeker should be implemented on smaller image subregions, possibly horizontal strips (range zones). Secondly, the required post-processing step of extracting connected components is rather difficult to implement in real-time hardware. However, the image smoothing algorithm [29], upon which the mode seeker is based would be an excellent preprocessor for any detection algorithm. An analysis of its hardware implementability is not yet complete.

CONCLUSIONS

The following conclusions were drawn from the study.

- Mode seekers, double window filters, and spoke filters all show promise as targets detectors. Border followers and possibly mode seekers (if hardware implementation can be worked out) are fair segmentors.
- No existing computer architecture appears appropriate for the real-time implementation (on one or two circuit boards) of all detection/segmentation algorithms. A special architecture is required for the efficient implementation of each particular algorithm.

It was originally intended to test an artificial intelligence (AI) approach to improving algorithm performance. This appears to be much more difficult than initially thought. The investigation has not revealed any real-time target acquisition system which uses AI concepts. It is concluded that for the present efforts are better directed to the use of available information, such as range and feedback data. At a later time, an attempt can be made to incorporate a simple knowledge base and a limited reasoning ability. Several possible examples might be:

- Ground vehicles are likely to be located below the horizon.
- Targets of the same type often appear in groups, sometimes in moving columns on or near roads.
- Gathered intelligence indicating presence of different target types is often available.

A STRATEGY

A sound strategy is to precede detection by preprocessing filters designed to supress noise and "bring out" targets. The Narayan filter may be of use here. This should be followed by an initial detection algorithm which is cheap to implement in hardware, but which is not necessarily a "star" performer. This algorithm should be run at a low threshold (i.e., high false alarm rate) to make sure that most targets are detected. This should be followed by a more expensive detection/segmentation algorithm which will operate only on the initial detections. Performance should be improved by target tracking, frame-to-frame integration of extracted data, and decision smoothing.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Army under the DARPA Image Understanding Program. The Army program monitor is Dr. George Jones of the Night Vision and Electro-Optical Laboratory, Ft. Belvoir, Virginia. The University of Maryland program director is Prof. Azriel Rosenfeld. The Westinghouse project manager is Dr. Glenn Tisdale. Prof. Rosenfeld and Dr. Tisdale reviewed a preliminary version of this manuscript and suggested improvements.

The FLIR data base was compiled by Rich Kroupa of Westinghouse. The Westinghouse border follower was developed by Arden Helland.

The experimental work at the University of Maryland was conducted by Les Kitchen, Ralph Hartley, and Cheng-ye Wang. The relaxation software was written by Russ Smith [25]. The mode seeking software was written by Krishnan Narayanan.

Dr. Marty Woolfson of Westinghouse suggested the double window probability density filter. Our evaluation of the hardware implementability of the various algorithms benefited from discussions with Westinghouse hardware designers Arden Helland, John Ferma, Jim Ross, and Rod Fletcher.

Special thanks goes to Dr. Walton Perkins and the staff of the General Motors Research Lab for testing our data base on their system.

REFERENCES

1. A.S. Politapoulos [Ford Aerospace], "An Algorithm for the Extraction of Target-like Objects in Cluttered FLIR Imagery," *AESS Newsletter,* Nov. 1980, pp. 23-31.

2. M. Burton and C. Benning [Texas Instruments], "A Comparison of Imaging Infrared Detection Algorithms", presented at SPIE 25th Annual Technical Symposium, Aug. 1981, San Diego.

3. (D.L. Milgram and A. Rosenfeld), "Algorithms and Hardware Technology for Image Recognition", Second Semi-Annual Report: (Nov. 1, 1976 - April 30, 1977), DARPA Order 3206, Contract DAA G53-76-0138. University of Maryland Computer Science Center, College Park, MD.

4. L.G. Minor and J. Sklansky, "Detection and Segmentation of Blobs in Infrared Images", *IEEE Trans Systems, Man, and Cyber,* March 1981, pp. 216-232.

5. C. Kimme, D. Ballard, and J. Sklansky, "Finding Circles by an Array of Accumulators", *Comm. ACM* 18, Feb. 1975, pp. 120-122.

6. R.L. Frey, C.D. Nealy, L.M. Rubin, and R.M. Wilcox, "ATAC Autocuer Modeling Analysis", Hughes Aircraft Co., Image Processing Lab., Culver City, CA, (for U.S. Army Night Vision and Electro-Optics Lab., Fort Belvoir, VA., report number FR-80-70-1325), January 1981.

7. H. Elliott, D.B. Cooper, and P. Symosek, "Implementation, Interpretation and Analysis of a Suboptimal Boundary Finding Algorithm", Proc. IEEE Pattern Rec. and Image Proc. Conf. (Chicago, Aug. 6-8, 1979), pp. 122-129.

8. A. Martelli, "An Application of Heuristic Search Methods To Edge and Contour Detection", *Comm. ACM,* 19, Feb. 1976, pp. 73-83.

9. P.S. Schenker, and D.B. Cooper, "Fast Adaptive Algorithms for Low-Level Scene Analysis: The Parallel Hierarchical Ripple Filter", SPIE Proc., Vol. 252, Smart Sensors II, Aug. 1980, pp. 113-123.

10. A. Rosenfield, "Extraction of Topological Information from Digital Images", Computer Science Dept. Tech. Report 547, Univ. of Maryland, College Park, June 1977.

11. Y. Yakimovsky, "Boundary and Object Detection in Real World Images", *J. ACM* 23, 1976, pp. 599-618.

12. A. Rosenfield, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations", *IEEE Trans. Systems, Man, and Cyber. 6,* June 1978, pp.420-433.

13. R. Kirby, "A product rule Relaxation Method", *Computer Graphics and Image Proc. 13*, June 1980, pp. 159-189.

14. S. Peleg, "A New Probabalistic Relaxation Scheme", *IEEE Trans. Pattern Analy Mach. Int.* 2, July 1980 pp. 362-368.

15. W.A. Perkins, "Area Segmentation of Images Using Edge Points", *IEEE Trans. Pattern Analysis Mach Int.* 2, Jan. 1980, pp. 8-15.

16. W.A. Perkins, "Simplified Model-Based Part Locator", Proc. Fifth Intl. Conf. on Pattern Recognition (Miami Beach, FL, Dec. 1-4, 1980), pp. 260-263.

17. B. J. Schachter and G. E. Tisdale, "Evaluation and Real-Time Implementation of Image Understanding Algorithms", Proc. Image Understanding Workshop, April 1981, pp. 178-183.

18. R.V. Hogg and A.T. Craig, *Introduction to Mathematical Statistics*, MacMillian, 1970.

19. A.J. Danker and A. Rosenfeld, "Blob Detection by Relaxation", *IEEE Trans. Systems, and Man, and Cyber.* 3, Jan. 1981, pp. 79-92.

20. D. Milgram, "Region Extraction Using Convergent Evidence", Proc. Image Understanding Workshop, April 1977, pp. 58-64.

21. D. Milgram, "Progress Report on Segmentation Using Convergent Evidence", Proc. Image Understanding Workshop, Oct. 1977, pp. 104-108.

22. T. Silberberg, S. Peleg, and A. Rosenfeld, "Multiresolution Pixel Linking for Image Smoothing and Segmentation", in Proc. Image Understanding Workshop, April 1981, pp. 33-38.

23. A Rosenfeld, A. Danker, and C. R. Dyer, "Blob Extraction by Relaxation", Proc. Image Understanding Workshop, April 1979, pp. 61-65.

24. A. Rosenfeld, "Some Uses of Pyramids in Image Processing and Segmentation", Proc. Image Understanding Workshop, April 1980, pp. 112-115.

25. R.C. Smith, "A General Purpose Software Package for Array Relaxation", University of Maryland, Computer Science Center, TR-839, Dec. 1979, College Park, MD.

26. T. Silberger, S. Peleg, and A. Rosenfeld, "Multiresolution Pixel Linking for Image Smoothing and Segmentation", University of Maryland Computer Science Center, TR-977, Nov. 1980, College Park, MD.

27. K.A. Narayanan, S. Peleg, A. Rosenfeld, and T. Silberberg, "Iterative Image Smoothing and Segmentation by Weighted Pyramid Linking", University of Maryland Computer Science Center, TR-989, Dec. 1980, College Park, MD.

28. P. Burt, T.H. Hong, and A. Rosenfeld, "Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation," University of Maryland Computer Science Center, TR-927, Aug. 1980, College Park, MD.

29. K. A. Narayanan and A. Rosenfeld, "Image Smoothing by Local Use of Global Information", University of Maryland, Computer Science Center, TR-1006, Feb. 1980, College Park, MD.